erschienen:
Pietsch, W.; Krams, B. (Hrsg.): Vom Projekt zum Produkt. GI-EditionLecture Notes in Informatics P-178, Bonn, S. 61-78.
Proceedings Fachtagung GI-Fachausschuss *Management der Anwendungsentwicklung und -wartung* im Fachbereich Wirtschaftsinformatik (WI-MAW), Aachen, Dez. 2010.

Campus-Management-Systeme

– Vom Projekt zum Produkt –

Markus Bick[1)], Thomas Grechenig[2)], Thorsten Spitta[3)]

[1)] ESCP Europe Wirtschaftshochschule Berlin,
Wirtschaftsinformatik

[2)] TU Wien, Softwaretechnik und Interaktive Informatik

[3)] Universität Bielefeld, Angewandte Informatik/Wirtschaftsinformatik

markus.Bick@escpeurope.de
thomas.Grechenig@inso.tuwien.ac.at
thSpitta@wiwi.uni-bielefeld.de

Abstract: Immer mehr Hochschulen führen Campus-Management-Systeme ein, um der vorwiegend aufgrund des Bologna-Prozesses komplexer werdenden Prozesse gerecht werden zu können. Vor diesem Hintergrund befasst sich der vorliegende Beitrag mit zwei Fragen. Erstens: *Was ist überhaupt ein Campus-Management-System?* Gehören alle Anwendungssysteme einer Hochschule dazu? Zweitens: *Wie können aus den in vielen Hochschulen laufenden Pilotprojekten Standardsysteme werden?* Nach einer Bestandsaufnahme und Bewertung der sichtbaren Ansätze bzw. Projekte werden mögliche Wege zu Standardprodukten aufgezeigt. Dabei wird den Hochschulen eine Checkliste zur Prüfung bereits existierender Lösungen an die Hand gegeben, um deren Bewertung im Hinblick auf eine mögliche Auswahl zu erleichtern .

1 Das Problem

Die im Zuge des Bologna-Prozesses zunehmende Umstellung deutschsprachiger Studiengänge auf sog. konsekutive Abschlüsse macht auch für diejenigen Organisationen „betriebliche“ Informationssysteme notwendig, die bisher glaubten, darauf weitestgehend verzichten zu können: die Hochschulen. Für eine effektive Steuerung und Planung der Lehre und der damit verbundenen Prüfungen werden jetzt dringend Systeme gebraucht, die die relevanten Vorgangsdaten der korrespondierenden Prozesse innerhalb der Hochschulen einheitlich festhalten bzw. „buchen“ und dabei zu Führungsinformationen verdichten. Da mit der mehr oder minder amtlich erzwungenen Umstrukturierung auch eine Öffnung der Studiengänge und Fakultäten einher geht, ist es auch nicht mehr möglich, dass Fachbereiche weiter mit eigenen Insellösungen arbeiten.

Für Organisationen wie Unternehmen oder Behörden ist es seit langem selbstverständlich und gehört auch zum Lehrbuchwissen (z. B. [Me04]), dass man für eine erfolgreiche Unternehmensführung u. a. auch Systeme mit einer einheitlichen Datenbasis betreibt oder anstrebt. Auch für Hochschulen ist diese Erkenntnis nicht neu ([Kü97, S.422ff.],

[SK95], [Sp97], [We96]), die Umsetzung wurde aber lange ignoriert. So kam es, dass erst in jüngster Zeit Hochschulen wie z. B. die Universität Hamburg [UHH10] oder die Technische Universität München [TUM08] millionenschwere Ausschreibungen starteten, um ein *Campus-Management-System* (CM-System) zu implementieren. Insbesondere die Prozesse in der Lehre sind ohne einheitliche Buchungssysteme vor allem für große Fakultäten nicht mehr zu beherrschen. Diese Prozesse sind in den Hochschulen von heute geprägt durch

- eine extreme Erhöhung der Anzahl zu buchender Vorgänge (studienbegleitendes Prüfen),
- den Zwang zur Verkürzung der Prozesslaufzeiten mit verbindlichen Endterminen, insb. beim Übergang vom Bachelor zum Master,
- den Wunsch der Hochschul- und Fakultätsleitungen nach hochwertiger, zeitnaher Information,
- die bessere Nutzung der durch die neuen Studienmodelle knapperen räumlichen Ressourcen (stark gestiegene Anwesenheitsquote),
- hohe Anforderungen an den Schutz sensibler, personenbezogener Daten (elektronische Prüfungsakten, Auskünfte und Bescheinigungen über Internet-Technologien).

Seit mehr als 30 Jahren bietet in Deutschland die HIS Hochschul-Informations-System GmbH – deren Gesellschafter alle Bundesländer und zu 25% der Bund sind – sogenannte „Standardsysteme“ im Hochschulbereich an. Die HIS-Software wird dabei nicht als releasefähige Standardsoftware betrieben und dies wohl auch in Zukunft nicht werden können. Eine gewisse amtlich verordnete Benutzung hat dazu geführt, dass sich in diesem wichtigen Bereich keine privatwirtschaftliche Konkurrenz entwickeln konnte. Neben softwaretechnischen Defiziten wird vor allem die institutionelle Unzuverlässigkeit von HIS bemängelt [Sp97], die seit Jahren mehr verspricht[1] als die vorhandenen Softwareentwicklungs- und Wartungskapazitäten jemals zu leisten im Stande sein können (s. bspw. [TUM10]).

Vor diesem Hintergrund sind die großen Anstrengungen zu bewerten, die fast alle Hochschulen im deutschsprachigen Raum unternehmen, um die neu modularisierten Studiengänge weitestgehend störungsfrei abzuwickeln. Dabei überwiegen Eigenentwicklungen, wie in Bielefeld [Br+09] oder Göttingen [Ra09].

Dies führt zu zwei Kernfragen, die dieser Beitrag beantworten will:

- Was ist ein Campus-Management-System?
- Wie könnten aus den Pilotprojekten in den Hochschulen Produkte entstehen?

[1] Das gut zu lesende Papier [De+09] gehört in die Liste solcher Versprechungen. Sie wurden bereits auf der zitierten GI-Jahrestagung 1997 in Aachen gemacht [Sp97].

Ziel ist es, diese Fragen in den beiden zentralen Abschnitten des vorliegenden Beitrags zu klären. Da die Frage der Wirtschaftlichkeit solcher Systeme unseren Rahmen und Fokus überschreiten würde, sei hierzu auf [Be09] oder [Sp+10] verwiesen.

2 Was ist ein Campus-Management-System?

Es ist allgemeiner Konsens und sogar in einem wohl weltweit erfolgreichen IT-Produkt deutschen Ursprungs manifestiert, was ein sog. „Betriebliches Informationssystem" bzw. „Enterprise Resource Planning (ERP)-System" ist (vgl. [Fi99] oder [Me04]). Die Bezeichnungen der „Module" etwa innerhalb des R/3- Systems der SAP AG folgen im Wesentlichen den üblichen Begriffen betriebswirtschaftlicher Funktionen [AR99, S. 36ff.]. Das Modell dieser Funktionsbezeichnungen ist der Industriebetrieb.[2]

Im Folgenden werden die generalisierbaren „betrieblichen" Funktionen einer Hochschule daraufhin untersucht, ob sie sich durch vorhandene Standardsoftware abdecken lassen oder ob sie hochschulspezifisch sind und damit auch spezifische Softwarekomponenten benötigen, die man *Campus-Management-System* nennen könnte.

2.1 Grundfunktionen und Prozesse in Hochschulen

Eine Hochschule bietet als Dienstleister einer Gesellschaft mit dem Prozess *Lehre* Qualifikationen von Absolventen und durch den Prozess *Forschung* Erkenntnis. Dies erscheint einfach, ist es aber aufgrund der hier vorgenommenen Abstraktion nur auf den ersten Blick. In der Mikrostruktur geben die verschiedenen Disziplinen dem jeweiligen Prozess ein vielfältiges Gesicht. Unterstützend gibt es einen weiteren Prozess, der ebenfalls stark disziplinabhängig ist und den wir dem üblichen Sprachgebrauch folgend *Bibliothek* nennen.

Nun ist es nicht Ziel dieses Beitrags, die genannten drei Grundfunktionen einer Hochschule in ihren Details zu diskutieren. Der Fokus muss vielmehr auf den Funktionen liegen, in denen sich standardisierbare Prozesse mit hohen Wiederholungsraten finden, die nach einer Effizienz- und Sicherheitserhöhung verlangen. Ein Beispiel für *dringenden* Unterstützungsbedarf wäre das schnelle und sichere Eintragen der Noten einer Veranstaltung mit 800 Teilnehmern.

Ein Beispiel für *nicht dringenden* Bedarf wäre die tägliche Ausleihe und Rückgabe von 5000 Büchern. Die Softwareunterstützung für Bibliotheken hat eine lange Tradition und gilt im Wesentlichen als zufrieden stellend.

Obwohl die Unterstützung der Forschung zunehmend auf Antragsverfahren und Drittmittel abzielt, die sich z. T. durch IT unterstützen lassen, werden wir diesen Prozess hier nur am Rande streifen. Es hat sich nichts Grundsätzliches geändert, zumal die Fallzahlen sehr viel geringer sind als in der Lehre. Zudem beruhen die Unterstützungsmöglichkeiten weitestgehend auf Office-Systemen oder sind derart fachspezifisch, dass sich eine Ein-

[2] Siehe für die Systeme Navision und R/3 bspw. [SB08, S. 131].

bindung in ein zentrales System verbietet. Als Service eines Campus-Management-Systems wünschenswert sind sicherlich eine für die Hochschule einheitliche Projekt- und Publikationsdatenbank.

Als spezifischer Bedarf bleibt somit eine wirksame, für jede Hochschule einheitliche, nachhaltig zur Verfügung stehende IT-Unterstützung des Prozesses *Lehre* (Abschnitt 2.2). Dieser ist umso schwergewichtiger, als Bachelor (BA) und Master (MA) äußerst repetitiv und von hoher Benutzer-Komplexität gekennzeichnet sind: Alle Studierenden und Lehrenden sind Stakeholder. Damit handelt es sich bei Entwicklung und/oder Einführung eines Campus-Management-Systems um ein sehr ernst zu nehmendes IT-Projekt [Gr+10, S. 91].

Ergänzend kommen die Unterstützungsprozesse der *Verwaltung* hinzu, wie *Personalwesen*, *Rechnungswesen/Controlling* oder *Beschaffung*. Diese Prozesse lassen sich gut durch Standardsoftware abdecken.

Dies führt zu der in Abbildung 1 dargestellten Sicht zentral zu betreibender Anwendungssysteme einer Hochschule, in der das Campus-Management-System zunächst nur grob angedeutet ist (s. auch [Gr+10, S. 133]).

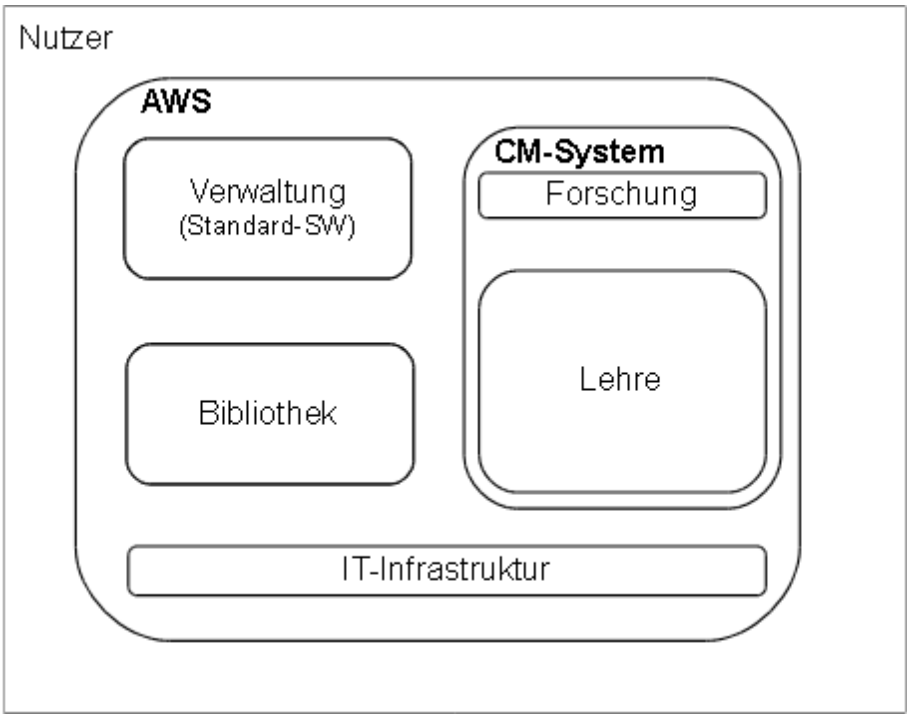

Abbildung 1: Zentrale Anwendungssysteme (AWS) einer Hochschule.

In dieser Sicht zählen Anwendungssysteme, die den einzelnen Arbeitsplatz oder Arbeitsgruppen unterstützen, zur *IT-Infrastruktur*. Nicht explizit eingezeichnet ist die Funktio-

nalität *E-Learning*. Sie wird erst bei einer genaueren Betrachtung des Prozesses Lehre sichtbar.

2.2 Der Prozess Lehre

Der Lehrprozess zerfällt in zwei grundsätzlich verschiedene Teilprozesse, die *Administration* und die *Durchführung*[3]. Die *Administration* ist gut standardisierbar und muss für alle Lehrveranstaltungen aller Disziplinen strukturell gleich sein. Dies gilt für alle Lehr- und Prüfungsformen, von der betreuten Hausarbeit bis zur Massenvorlesung. Die Prozessschritte sind, selbst bei sukzessiver Bewertung in kleinen Seminaren ohne explizite Prüfung:

Organisation → Durchführung → [Prüfung] → Bewertung.

Abbildung 2 zeigt daraus den Teilprozess einer expliziten Prüfung.

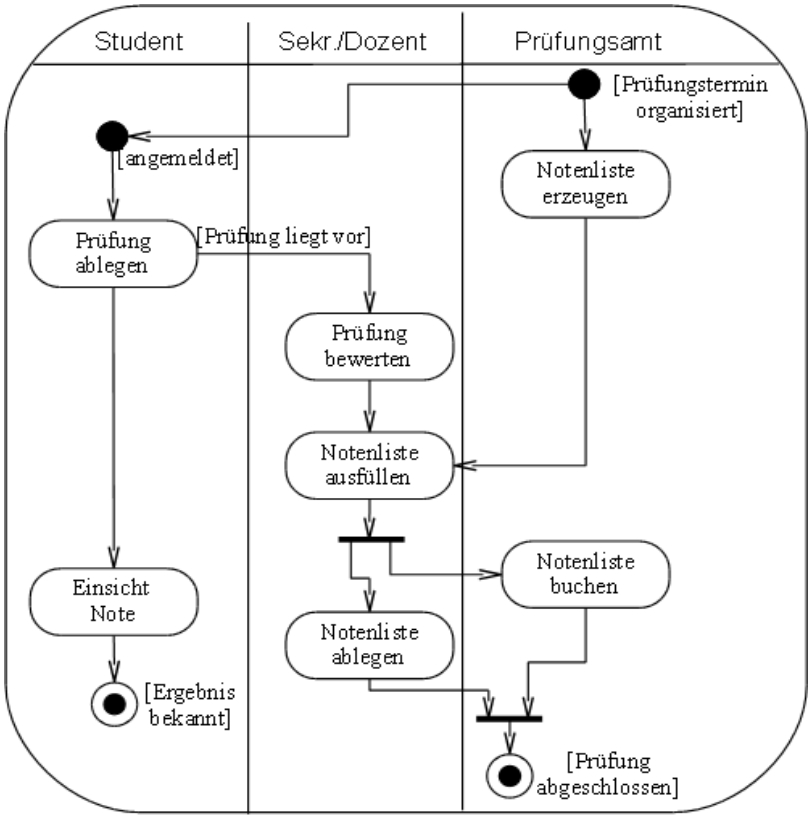

Abbildung 2: Der Teilprozess *Prüfung* aus dem Administrationsprozess der Lehre

[3] Die inhaltliche Vorbereitung durch den Dozenten behendeln wir nicht zuletzt aus Gründen der Selbstbeschränkung hier nicht.

In einer üblichen deutschsprachigen Universität läuft dieser Prozess zum Semesterende rund zweitausend Mal ab, was zu rund 30 Mio. Buchungen führt.[4] Solche Mengengerüste und die damit verbundene Datensicherheit und -konsistenz kann nur noch durch ein entsprechendes Anwendungssystem geleistet werden, das auf einer zentralen, transaktionssicheren und gut skalierenden Datenbank aufsetzt.

Der Teilprozess *Durchführung* kennt nur noch die Akteure Dozent(en) und Studierende. Er ist wenig standardisierbar und hängt von der Anzahl der Studierenden, dem Aufwand des Lehrenden und auch von der Fachdisziplin und dem Lehrinhalt ab. In diesem Teilprozess kann E-Learning-Software hilfreich sein. Wenn sie eingesetzt werden soll, ist zu fragen, wo es Überschneidungen mit administrativen Teilen des Lehrprozesses gibt. Wir kommen auf sie zurück, wenn die strukturbildenden Bestandteile administrativer Software im folgenden Abschnitt an Hand des Beispiels *Person* besprochen sind.

2.3 Teilsysteme administrativer Software und ihre Schnittstellen

Die Funktionen administrativer Anwendungssysteme werden durch eine einheitliche Datenbasis integriert [AR99, S. 5], [Gr+10, S. 161]. Die Struktur der Datenbasis bestimmt maßgeblich die Aufteilung von Systemen in Teilsysteme, die über möglichst wenige Schnittstellen kommunizieren sollen. Dies ist eine alte Erkenntnis, die schon in einer 1988 abgeschlossenen Habilitationsschrift als Kernaussage enthalten war [Sp89, Kap. 6].

Was heißt „wenige Schnittstellen“? Man will ein Campus-Management-System als selbständigen Kern definieren, der nicht mit unnötiger Kommunikation oder Spezialfunktionen belastet ist. Das Konstruktionsprinzip hierfür ist, dass originäre Daten immer nur in genau einem Teilsystem erzeugt bzw. geändert werden dürfen: Nur *ein* System kann der Owner eines Datentyps sein. Lange bevor dieses Prinzip unter dem Namen *Objektorientierung* bekannt wurde, war es bereits intuitiv in den 80er Jahren im System R/2 der SAP berücksichtigt worden.

Damit dies für ein Campus-Management-System konkreter wird, zeigt Abbildung 3 das grundlegende Datenmodell eines solchen Systems. Die vielfältig möglichen abgeleiteten Daten sind nur beispielhaft dargestellt. Auch die Komplexität der hier nur grob skizzierten Klassen, insb. die komplexe Klasse *Studiengang*, kann hier nur angedeutet werden.

Am Datenmodell der Abbildung 3 kann man auch die Schnittstellenproblematik der Anwendungssysteme aus Abbildung 1 zeigen, die wir im Folgenden exemplarisch am Beispiel der Personaldaten diskutieren.

[4] Rechenbeispiel: (250 Dozenten * 4 + 1000 wissenschaftliche Mitarbeiter *1) * 15000 Studierende.

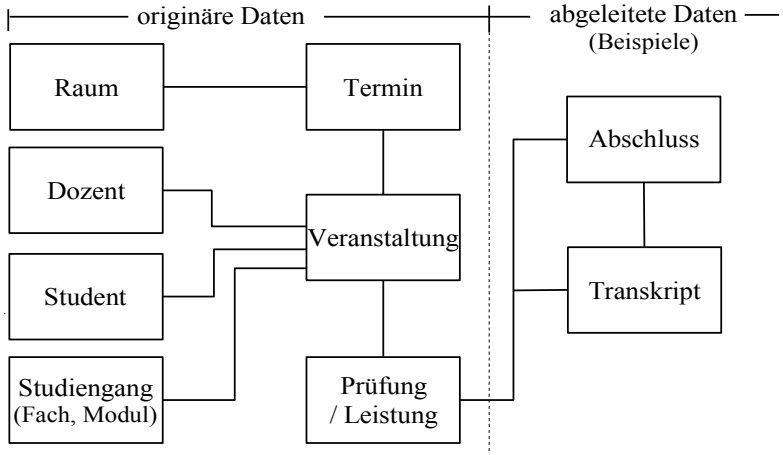

Abbildung 3: Grobes Datenmodell eines Campus-Management-Systems

In Lehrbüchern findet man oft eine Klasse *Person*, spezialisiert in die erbenden Klassen *Dozent* und *Student*. Damit hätte man bereits den ersten schweren Entwurfsfehler gemacht, der zwei verschiedene Systeme unnötig verkompliziert. Owner von *Dozent* darf nämlich *nicht* das Campus-Management-System sein, dies ist hier beispielsweise SAP/HR (Human Resources). Das Campus-Management-System braucht eine Importschnittstelle aus HR für diejenigen Attribute, die es für seine Klasse *Dozent* benötigt. Ohne eine solche Klasse wäre kein Campus-Management-System arbeitsfähig. Das HR-Modul wiederum würde geradezu „missbraucht" und müsste auch unnötigen Schnittstellenverkehr[5] verkraften, würde es mit Studierenden-Daten belastet.

Auch alle anderen Teilsysteme (Bibliothek, Forschung, IT-Infrastruktur, E-Learning) dürfen keine Funktionen enthalten, die Personaldaten verändern. Dem gegenüber macht es sehr viel Sinn, die Klasse *Student* als wichtigsten Stakeholder des Systems ausschließlich im Campus-Management-System anzusiedeln. Wenn nicht erkannt wird, dass *Bewerber* und *Alumni* nur Rollen eines komplexen Datentyps *Student* sind, erzeugt das weitere Entwurfsfehler, die unnötige Schnittstellen provozieren. Dass eine funktionsorientierte Modularisierung (isolierte Systeme für Bewerber, Aktive und Alumnis) sehr negativ ist, wissen wir schon seit dem Bahn brechenden Artikel von Parnas [Pa72].

Aus dieser beispielhaften Diskussion über die Minimierung von Schnittstellen lässt sich eine grundlegende technische Anforderung an jedes Campus-Management-System ableiten: Ein CM-System muss offenen Schnittstellen-Standards genügen. Dies gilt als „fundamentale Best Practice" [Gr+10, S. 216]; Bick/Börgmann bezeichnen *Platform Independence* sogar als 'Knockout Criterion' [BB09, S. 111] für die Entwicklung und/oder Auswahl derartiger Systeme. Die große Bedeutung dieses Prinzips zeigt im folgenden Abschnitt ein Blick auf die Funktionen eines Campus-Management-Systems.

[5] So war beispielsweise die Schnittstelle zwischen den HIS-Systemen SOS (Studentendaten) und POS (Prüfungsverwaltung) in der Fakultät Wirtschaftswissenschaften der Universität Bielefeld lange problembehaftet. Wegen ihrer hohen Störungsanfälligkeit wurde sie möglichst nur zweimal im Semester benutzt.

2.4 Funktionen eines Campus-Management-Systems

Ein Campus-Management-System muss die Ressourcen seiner Stakeholder für den kombinatorisch hoch komplexen Prozess Lehre effizient und problemorientiert verwalten. Problemorientierung heißt vor allem, dass Pläne praktisch ausführbar sein müssen, insb. Studienpläne. *Studierbarkeit* wirksam zu unterstützen, muss eines der wichtigsten Ziele eines Campus-Management-System sein [St+07, S. 20]. Die Ressourcen, die hierfür miteinander in Einklang gebracht werden müssen, sind *Zeit*, *Raum* und *Menschen*. Dies führt neben der Grunddatenverwaltung von Studierenden, Raum und Studiengang (s. o.) auf die Grundfunktionen *Raumplanung*, (individuelle) *Stundenpläne* und *Prüfungsverwaltung*. Diese Grundfunktionen bedingen natürlich eine einheitliche Darstellung des Lehrangebots. Alle diese Funktionen basieren auf Vorgangsdaten[6], die von Benutzern in Prozessen gebucht werden. Aus den originären Grund- und Vorgangsdaten werden für eine Vielzahl von Stakeholdern abgeleitete Daten erzeugt, umgangssprachlich oft „Informationen“ genannt. Jetzt kommen die Führungskräfte aus Fachbereichen und Universitätsleitung hinzu, meist unterstützt durch eine Controllingfunktion. Grechenig et al. zeigen die sehr umfangreiche Tabelle der Stakeholder eines Campus-Management-Systems [Gr+10, S. 196f.], Bick/Börgmann eine detaillierte Liste der 'Areas of Responsibility' , z. B. Grunddatenverwaltung, Studentendaten-Pflege und Veranstaltungsverwaltung [BB09, S. 108].

In den Funktionen können sich Überschneidungen mit E-Learning-Systemen ergeben, bspw. bei der Bereitstellung von Lerninhalten oder der Betreuung von Gruppen im Rahmen großer Veranstaltungen. Hier sollten zwei Zurechnungskriterien für das Campus-Management-System oder für ein entsprechendes Spezialsystem gelten.

1. *Generalisierbarkeit* dürfte bei der Verteilung von Lernmaterialien gegeben sein, denn sie ist für fast alle Veranstaltungstypen aller Disziplinen erforderlich. Dies ergibt sich schon aus der notwendigen Nachbarschaft zu den inhaltlich/organisatorischen Veranstaltungsbeschreibungen.

2. *Bezug zu den Grunddaten*: Eine Gruppeneinteilung für Tutorien bzw. Aufgabengruppen oder eine Zuteilung zu kapazitätsbeschränkten Lehrveranstaltungen umfasst die Studierenden einer Veranstaltung und ggf. Tutoren als Lehrende und damit wesentliche Grunddaten. Ergänzungssysteme mit starkem Bezug zu Grunddaten schaffen als isolierte Systeme mehr Probleme als sie lösen.

Dagegen muss es möglich sein, funktionale Bausteine z. B. aus vorhandenen Methodenbibliotheken zu nutzen, etwa für Überschneidungsberechnungen zwischen Lehrangebot und vorläufigen Stundenplänen. Es macht wenig Sinn, ein zentrales System mit der Re-Implementierung spezieller Algorithmen zu überfrachten, wenn es dafür Standardprogramme gibt. Für die Nutzung solcher Zusatzfunktionen benötigt man unbedingt offene Schnittstellen.

Allein wegen der großen Zahl der wichtigsten Stakeholder eines CM-Systems, der Studierenden, gibt es eine Basisfunktion, die in kommerziellen Systemen keine ganz so

[6] Auch der Begriff *Bewegungsdaten* ist üblich; Grunddaten heißen in einem solchen Kontext *Stammdaten*.

große Bedeutung hat wie in einem Campus-Management-System: *Kommunikation*. Warum? Der „Kunde“ ist der Teilnehmer einer Veranstaltung. Er befindet sich *im* System und nicht wie sonst in der Umwelt eines Systems. Jeder Dozent muss die Möglichkeit haben, mit genau den Studierenden zu kommunizieren, die für seine Veranstaltung eingetragen sind. E-Mail ist hierfür ein bewährtes Standardsystem, bei dem allerdings hohe Sicherheitsanforderungen etwa an Verteiler gestellt werden müssen. Dies wird in der Bielefelder Entwicklung seit Jahren mit Erfolg genutzt [St+07, S 24].

Es versteht sich, dass alle Funktionen über intuitiv zu erlernende, offene Benutzerschnittstellen erreichbar sind, Buchungen und Auskünfte mit personenbezogenen oder geheimhaltungsbedürftigen Daten aber in geschütztem Umfeld stattfinden.

2.5 Zwischenfazit

Ein Campus-Management-System muss – analog zu betrieblichen Informationssystemen – auf einer zentralen Datenbasis aufsetzen, denn es ist ein primär buchendes System mit leistungsfähigen Auskunft- und Auswertungsfunktionen. Seine Teilsysteme (sog. „Module“) sollten sich um die Grunddaten gruppieren, von denen Dozent, Studierende, Raum und Studiengang die wesentlichen sind. Einer der wichtigsten Vorgangsdatentypen ist *Veranstaltung*. Alle buchenden und auswertenden Operationen auf diesen Datentypen müssen Teil des Systems sein. Eine Ausnahme bilden die Grunddaten entlohnter Personen (Datentyp *Dozent*), die außerhalb des Systems gepflegt werden müssen, denn es macht keinen Sinn, allgemeine Personalfunktionen in ein Campus-Management-System zu implementieren. Hier benötigt das System aus Effizienzgründen die relevanten Daten mit nur *lesenden* Operationen.

Mit den Datentypen eng verbundene Funktionen sollten Teil des Systems sein, Berechnungsfunktionen ohne Gedächtnis sollten aus Standardbibliotheken bezogen werden können. Daher sind ergänzende Funktionen, die selbst keine redundanten Grunddaten pflegen, gerade auch aus dem Repertoire des E-Learnings zu nutzen.

Offene Schnittstellen sind für ein Campus-Management-System ein absolutes Muss, um evolutionäre Erweiterungen und Ergänzungen zu ermöglichen, die im deutschsprachigen Bildungswesen sehr wahrscheinlich sind.

3 Von Pilotprojekten zu Standardprodukten

Nach der vorangegangenen Bestandsaufnahme werden im Folgenden Anforderungen an Standardprodukte formuliert, um danach der Frage nachzugehen, welche Bedingungen erfüllt sein müssen, damit ein nachhaltig wartbarer Standard entstehen kann.

3.1 Landschaft der Standardprodukte

Bick/Börgmann haben aktuell eine europaweite Übersicht von 22 Campus-Management-Systemen erstellt [BB09, S. 109]. Von ihnen sind für den deutschen Sprachraum vor allem die in Tabelle 1 gezeigten fünf Systeme[7] interessant:

Nr	Name	Pilotanwendung	Anmerkung	Hersteller in
1	Campus-Management SAP	FU Berlin	offenbar von SAP keine Fortsetzung	Walldorf
2	Campus Net (Datenlotsen)	Uni Hamburg	Echtbetrieb mit Störungen[8]	Hamburg
3	CampusOnline	TU München	Noch kein Echtbetrieb	Graz / Öst.
4	CAS Campus	RWTH Aachen	Standard (?)	Karlsruhe
5	HISinOne	s. www.hisinone.de	bisher nur Prototyp	Hannover

Tabelle 1: Angebot Campus-Management-Systeme im deutschsprachigen Raum

Von diesen fünf Systemen scheint das erste von SAP nicht fortgeführt zu werden. Zudem befinden sich das zweite und dritte CA-System nicht in dem Zustand, dass sie sich als Referenz eignen. Auch bestehen für das fünfte System vielleicht nicht mehr als Absichtserklärungen. Die Homepage von HISinOne.de enthält eine Landkarte von zehn Universitäten, darunter die RWTH Aachen, die auch Pilotanwender von CAS Campus war (Tabelle 1). Was korrekt ist, lässt sich leicht überprüfen. Obwohl die Idee einer breiten Nutzerbeteiligung für ein solches Vorhaben gut ist, sind Zweifel angebracht, ob die HIS GmbH ein so komplexes Projekt wird managen können. Da HISinOne eine zentrale Datenbasis nicht vermeiden kann, ist es verwunderlich, dass technisch belastbare Angaben zum System auf der Homepage fehlen.

Die CAS Software AG bietet im Rahmen ihrer Internetpräsenz als einziger Hersteller Hinweise auf „lebende“ Installationen, die im Internet einsehbar sind: RWTH Aachen, FH Aachen, TU Kaiserslautern, Universität Bochum. Von Gesprächspartnern, die andere Lösungen bevorzugen, wird angezweifelt, dass der Pilotanwender Aachen eine Standardlösung betreibt. Das lässt sich verifizieren oder widerlegen, indem man die Releasewechsel hinterfragt. Eine „Vorabschau“ im Internet kann natürlich keine seriöse Produkt- und Lieferanten-Überprüfung ersetzen. Mehr hierzu in Abschnitt 3.2.

In Hamburg (Universität und Hochschule für Angewandte Wissenschaften) wurde eine telefonische Umfrage eines der Autoren dieses Papiers mit insgesamt zehn Interviews von rund 30 Minuten durchgeführt. Die Mehrzahl der Gesprächspartner war unzufrieden. Das Projekt in der Universität Hamburg wurde 2006 begonnen und sollte nach dem Projektplan 2007 beendet sein. Bis in das Jahr 2009 hinein war der Betrieb von massiven Performance-Problemen begleitet, da offenbar weder das Rechenzentrum der Universität

[7] Drei weitere „deutsche Systeme“ aus der hier zitierten Übersicht sind nicht wirklich umfassende Systeme oder haben keine Referenzen und werden daher im Weiteren nicht berücksichtigt.
[8] s. http://www.info.stine.uni-hamburg.de/news.htm (5.8.2010)

noch der Hersteller Erfahrungen mit einer großen Datenbankinstallation hatte. Die Kosten für die Universität müssen hoch sein[9], werden aber strikt geheim gehalten.

Die Helmut-Schmidt-Universität in Hamburg (Universität der Bundeswehr) setzt Datenlotsen *nicht* ein, sondern HIS-Komponenten. Ihre Entscheidung von 2006 sei hier zitiert:

> „Die funktionalen und strukturellen Vorteile vor allem der Lösung „CampusNet" der Datenlotsen vermögen das finanzielle und planerische Risiko nicht aufzuwiegen." [HSU06, S. 6].

Wenn hier Negatives über einzelne Projekte geschrieben wird, dient das *nicht* dazu, die Produkte schlecht zu machen. Wir haben es hier mit Pilotanwendungen zu tun, in denen Störungen in Projekten dieser Größenordnung durchaus „normal" sind, d. h. sie lassen sich nicht vermeiden. Das war bei den ersten SAP-Installationen vor 25 Jahren nicht anders.

Was jedoch bedenklich stimmt, ist eine aggressive Vermarktungspolitik von Herstellern, während die Pilotinstallation noch nicht beendet ist. Dies beteiligt entweder den Kunden über Gebühr am Entwicklungsrisiko oder es überlastet den Anbieter und bedroht damit seine wirtschaftliche Existenz, auf die jeder Kunde bei nicht selbst entwickelter Software dringend angewiesen ist.

Die Produktlandschaft der *Campus-Management-Systeme* befindet sich in einem weit weniger ausgereiften Zustand als dies bei üblichen betrieblichen Anwendungssystemen der Fall ist. Welchen Anforderungen müssen Softwaresysteme genügen, die mit dem Ziel entwickelt wurden, daraus mehrfach einsetzbare Produkte zu machen?

3.2 Anforderungen an Standardprodukte

Ein Standard-Softwareprodukt muss neben der fachlich umfassenden Abdeckung einer Domäne vor allem zwei Eigenschaften haben, die es von qualitativ hochwertigen Einzelprodukten abhebt: Flexibilität und Nachhaltigkeit. Die beiden zuletzt genannten Eigenschaften sind eine Folge des Anspruchs, ein Problem allgemein gültig zu lösen. Die Abdeckung einer Domäne stellt sich nicht mit dem ersten Pilotprojekt ein, es können sogar Teillösungen entstehen, die sich als Irrweg herausstellen.

Unter Management-Gesichtspunkten lesen sich die Anforderungen so:

> „Damit das IT-Produkt in unterschiedlichen Einsatzfeldern bestehen kann, muss [es] ausgereift sein, d. h. es muss praktisch und technisch funktionieren." [HP09, S. 102].

Konsequenterweise betrachten wir hier das Attribut *ausgereift* sowie die *Funktionsfähigkeit* aus einer technischen Sicht. Die technischen Eigenschaften bestimmen maßgeblich die Investitionssicherheit, die der Hersteller festlegt und die den Kunden für die Einsatzdauer eines Produkts binden. Bei Softwareprodukten sind durch die starke Verzahnung mit der Organisation die Abhängigkeiten vom Produkt besonders groß.

Was bedeutet der Begriff *Nachhaltigkeit* in unserem Kontext? Standard-Softwareprodukte haben Lebenszyklen von 30 bis 40 Jahren. Ob 40 Jahre das Maximum ist, wissen wir

[9] Ein Zitat: „Der Vertrag ist für die Uni Hamburg eine Katastrophe."

mangels Historie noch gar nicht. In solchen Zeiträumen spielen sich dramatische Technologiesprünge ab, denen das Produkt folgen können muss. Es gibt viele Beispiele, bei denen das nicht gelungen ist, z. B. der Untergang der Fa. Nixdorf mit ihrem bis in die 90er Jahre sehr erfolgreichen Anwendungssystem COMET. Ein positives Beispiel für ein bekanntes Produkt, inzwischen über 30 Jahre alt:

> Das System R (Realtime) der späteren SAP AG war als R/2 für den Großrechnerbereich in einem Makro-Assembler programmiert, der auf IBM- und SIEMENS-Großrechnern ablauffähig war. Die Portabilität beruhte auf einer weitsichtigen Software-Architektur, damals vom Typ 2-Tier (Schicht). Die ersten Installationen gab es ungefähr 1980. 1986 – 1991 wurde die völlig neue 3 Tier-Architektur (R/3) nicht nur neu entworfen, sondern in der 4GL-Sprache ABAP IV neu implementiert. Heute entwickelt SAP in der Sprache Java.

Das Produkt R, heute ERP, hat auf Grund seiner Architektur und vieler anderer qualitätssichernder Maßnahmen die Flexibilität gehabt, der Technologie zu folgen. Nachhaltig war nicht der Code, sondern das gesamte Entwicklungsumfeld mit Quellcodeverwaltung, Versionsverwaltung, kontrollierten Konventionen, Frameworks, automatischen Tests und nicht zuletzt Offenheit gegenüber dem Kunden. Die Flexibilität gefördert haben sicher auch die im vorigen Abschnitt genannten offenen Schnittstellen.

Eine weitere wichtige Dimension der Flexibilität eines Campus-Management-Systems ist die Unabhängigkeit der Anwendungssoftware vom Betriebssystem [BB09, S. 111], einer der Erfolgsfaktoren schon von R/2. Dies ist besonders wichtig mit Bezug auf Client-Betriebssysteme und geht über das Betriebssystem hinaus. Ein Rich Client, die übliche Architektur der frühen 2000er Jahre [Gr+10, S. 205], darf in einer 3 Tier-Architektur nicht mehr vorkommen.

Damit nicht nur das gerne benutzte Erfolgsprodukt der Fa. SAP als alleiniges positives Beispiel stehen bleibt, sei kurz ein zentral betriebenes Einzelprodukt aus etwa der gleichen Zeit genannt, das sich am Markt als Standardsystem durchgesetzt hat, allerdings mit nur *einem* sehr mächtigen „Server“, einem Großrechenzentrum in Erding bei München.

> Die Entwicklungs- und Firmenhistorie des START-Systems für Reisebuchungen ist anschaulich in [Wiki10] geschildert, seitens der Entwicklung allerdings auf Endnutzer-Hardware beschränkt. Dies geht am hier relevanten, wesentlichen Erfolgsfaktor des START-Systems vorbei. Das System konnte sich von 1978 bis heute nur wegen seiner Software-Architektur evolutionär entwickeln und am Markt durchsetzen.

Denert hatte als Projektleiter diese Entwicklung stark befördert, indem er mit seinem Team erstmalig eine explizit konstruierte Architektur entwarf, die sich bis dahin bei Software eher implizit und beiläufig beim Programmieren ergeben hatte (vgl. [De79]). Auch die START-Software musste wie das R-System wegen mangelhafter Standardisierung in einer portablen Programmiersprache neu implementiert werden. Dem Erfolg des Produkts hat die konstruktive Pionierarbeit offenbar gut getan.

Von dem umfassenden Kriterienkatalog aus [BB09, S. 113], den Produkte erfüllen müssen, seien hier nur noch diejenigen zusätzlich genannt, die für Standardsysteme ein besonderes Gewicht haben: 'Data Privacy and Reliability', 'Integration with existing Systems'.

Wir fassen zusammen: Software, die mit dem Ziel *Standardprodukt* entwickelt wird, muss in besonderem Maße hohe technische Maßstäbe erfüllen, wie sie heute für die Entwicklung industrieller Qualitätssoftware üblich sind. Dies gilt bereits für die Anforderungen, ganz besonders für den Entwurf, die Implementierung und die den gesamten Prozess begleitende Qualitätssicherung, insb. den Test [Gr+10]. Die Entwicklung von Qualitätssoftware beginnt bei den Anforderungen, die exakt und für den Kunden nachvollziehbar dokumentiert werden müssen, weil sonst das Projekt niemals durch einen Abnahmetest beendet werden kann. Die systematische Qualitätssicherung muss für den Kunden nachvollziehbar sein, d. h. einsehbar. Zertifikate alleine genügen nicht.

Es werden also Mindestqualitäts-Anforderungen sowohl an die erste Version des Produkts als auch an die Leistungsfähigkeit des Herstellers gestellt.

3.3 Wie könnte ein Standard entstehen?

Die Analyse der letzten beiden Abschnitte beschreibt die folgende Situation:

- Die Nachfrage nach einem Standardsystem seitens der Hochschulen ist hoch.
- Es gibt nur die Vorstufe eines Angebots in Form einiger Pilotsysteme, über deren Qualität wenig bekannt ist.

Standardsysteme, die den Anforderungen der vorherigen Ausführungen genügen, müssen erst entstehen. Hierzu müssen Alternativen sowohl bei den Herstellern – die institutionelle Seite – als auch seitens des Produkts – die technische Seite – genauer betrachtet werden.

3.3.1 Institutionelle Alternativen

Wir zählen zunächst die Alternativen auf und bewerten sie dann.

1. *Garage* (Bsp.: Hewlett Packard, Microsoft)
2. *Platzhirsch Typ A* investiert (Bsp.: SAP AG)
3. *Platzhirsch Typ B* ändert sich (Bsp.: HIS GmbH)
4. *Pionier Typ A* mit Venture Capital (Bsp.: Datenlotsen, ca. 80 Mitarbeiter, Einprodukt-Unternehmen)
5. *Pionier Typ B* mit organischem Wachstum (Bsp.: CAS Software AG, ca. 200 Mitarbeiter; Mehrprodukt-Unternehmen)
6. *Kooperation* von Hochschulen (Bsp.: Land Mecklenburg-Vorpommern [MV04]).

Es ist keineswegs so, dass die zuvor genannten Alternativen einander ausschließen, im Gegenteil: Innovation lebt von Pioniergeist *und* Konkurrenz. Sicher ist nur, dass jeder potentielle Kunde – dies ist die oben festgestellte hohe Nachfrage – unbedingt benötigt:

- Präzise, selbst erstellte *Anforderungen*

- geeignetes *Personal*, falls das System später selbst betrieben werden soll
- eine exakte, auf den Anforderungen basierende *Ausschreibung*
- *fachjuristische Unterstützung* für Softwareverträge.

Der Typ *Garage* scheidet heutzutage ziemlich sicher aus, allein deshalb, weil sich wegen der hohen Nutzer-Komplexität keine realistisch große Pilotinstallation betreuen ließe. Pioniere wie die SAP-Gründer waren immerhin vier Personen und hatten mehr Entwicklungszeit, als heute zur Verfügung steht.

Vom realen *Platzhirsch Typ A* (SAP AG) wissen wir im Moment nicht, warum eine Produktlinie *Campus-Management-System* zur Zeit nicht weiter bearbeitet wird. Es könnte die unbefriedigende Ertragssituation in einem zu schwierigen Markt sein. An der Fähigkeit, nachhaltige Software zu entwickeln, mangelt es ganz sicher nicht.

Über den *Platzhirsch Typ B* (HIS GmbH) wurde in diesem Papier schon Verschiedenes gesagt. Es handelt sich um Bewertungen, die sicher nicht jeder teilt und teilen muss. Es gibt Hochschulen, die um die mangelnde Integration der Einzelpakete der Lösung wissen und dies bei der Bewertung der Alternativen in Kauf nehmen [HSS06, MV04]. Schließlich liefert die HIS GmbH seit Jahren benutzbare Software für viele Installationen, wenn auch das Anspruchsniveau bezüglich einer konsistenten Datenbasis reduziert werden muss. Die Autoren machen auch keinen Hehl daraus, dass sie bei den Herstellern Lösungen bevorzugen, die sich privatwirtschaftlich auf Dauer selbst tragen.

Die beiden verbliebenen privatwirtschaftlichen Hersteller gehören zur Klasse der *Pioniere*. Sie werden von den Autoren als Unternehmenstyp für die Herstellung eines Campus-Management-Systems bevorzugt, da hier eher echte Innovationen zu erwarten sind, wie sie zukünftige Produkte benötigen. Pioniere können nur Erfolg haben, wenn sie risikobereite Kunden finden. Dieses Risiko war das Land/die Universität Hamburg im Fall der Datenlotsen offenbar bereit einzugehen. Hinter dem Pilotprojekt stand als Nutzen die Verschmelzung zweier Hochschulen (Harburg und Hamburg). Allerdings muss sich gerade ein Pionier – anders als das in Hamburg wohl der Fall war – einer besonders kritischen Prüfung seiner Verfahren, seines Personals und vor allem der technisch-fachlichen Struktur des Pilotprodukts stellen.

Denkbar ist auch das Modell *Kooperation*. Es ist möglich, dass eine Hochschule mit einer gut funktionierenden Eigenentwicklung eine andere als kooperierenden Kunden aufnimmt. Die Variationsbreite der Art der Zusammenarbeit ist groß und könnte z. B. in die Gründung einer gemeinsamen Servicegesellschaft münden. Wenn man die Evolution der Software auch mit Personal der Partner-Hochschule vorantreibt, wäre ein Schritt zu einer Standardlösung getan, dem man weitere folgen lassen könnte. Diese Richtung deutet die Gründung kooperativer Rechenzentren an, eine mögliche Entwicklung zur wirtschaftlichen Verbesserung für Hochschulen insgesamt.

Für jeden Hersteller-Typ muss dessen Leistungs- und Überlebensfähigkeit dezidiert geprüft werden, aber auch seine Offenheit. Gerade in der Pilotphase – wir sehen Ihr Ende

frühestens nach der fünften vollständig abgeschlossenen Installation – ist eine auf Misstrauen gegründete Zusammenarbeit zwischen Hersteller und Kunde desaströs. Erst nach erfolgreich abgeschlossener Pilotphase, belegbar durch Referenzen, sollte man ein neues Softwareprodukt einen *Standard* nennen.

3.3.2 Technische Alternativen

Am Ende von Abschnitt 2 war als Campus-Management-System ein Anwendungssystem gefordert worden, das auf einer integrierten Datenbasis aufsetzt. Diese Anforderung ist nicht dadurch entkräftet, dass die HIS GmbH funktionierende Systeme mit separater Datenbasis pro Baustein liefert. Es herrscht Einigkeit, dass betriebliche Anwendungssysteme dieser Struktur als veraltet gelten [Me04]. Somit ist die bereits in Abschnitt 2 beschriebene technische Grobstruktur alternativlos.

Zudem fassen wir die schon in Abschnitt 3.2 diskutierten Eigenschaften noch einmal tabellarisch zusammen (Tabelle 2), da diese neben dem Anbieter maßgeblich für Ausschreibung und Entscheidung sind. Jeder Bieter muss die fachliche und technische Qualität seines Produkts offen legen. Dies geschieht durch stichprobenartige Inspektionen der Entwicklungs-Artefakte durch die ausschreibende Institution oder deren Beauftragte. Es ist selbstverständlich, dass Entwicklungsdokumente jeglicher Art dazu gehören und nicht nur Quellcode. Präsentationen sind keine hinreichende Entscheidungsgrundlage.

Als zentrale technische Eigenschaft für eine nachhaltige Entwicklung war die Architektur eines Softwareprodukts genannt und mit den Beispielen R und START untermauert worden. Architekturen werden schon lange über Frameworks implementiert [Sp89, Kap.10; Gr+10, S. 249ff.]. Da ein Framework sich über Skelettprogramme und ablauffähige Bausteine über das gesamte Produkt ausbreitet, kann ein unerprobtes Framework auch sehr nachteilig sein. Deshalb sind ergänzend auch die zu prüfenden Eigenschaften des Quellcodes in Tabelle 2 benannt.

Teilprodukt	Eigenschaft
fachlich	
Anforderungen	Differenziert, präzise & verständlich?
Datenmodell	Sachgerecht & detailliert genug (Attributtypen?)
Anwendungsfälle	Der Domäne angemessen; wesentlich?
Abnahme-Testfälle	Untestbare Anwendungsfälle?
technisch	
Grob-Architektur	Verständlich als Kommunikationsmittel innerhalb des Entwicklungsteams?
Datenbankentwurf	Sachgerecht & SQL-Dialekt-neutral?
Framework	Für die Entwickler verständlich dokumentiert?
Programmstruktur	Nachvollziehbar?
Schnittstellen	Systematisch?
Verständlichkeit	Ja oder Nein?
Fehlerbehandlung	Welches Konzept? Verhalten der Schichten gegenüber dem Nutzer?
Schichtenstruktur	Welche? Ist sie strikt eingehalten?
Client/Präsentation	Betriebssystem-unabhängiger Client?
Anwendung	Strikt getrennt von Präsentation?
DB-Zugriff	Kapselung des SQL-Dialekts?
Code	Nachvollziehbar in Struktur & Namenskonzept?
Variablennamen	Problemadäquat & systematisch?
Entwicklungs-Tickets	Existieren sie & werden sie benutzt?
Schnittstellen	Nachvollziehbar?
Programmaufbau	Verständlich?

Tabelle 2: *Vor* der Auftragserteilung zu prüfende Bestandteile eines neu entwickelten Softwareprodukts

Natürlich erwartet niemand, dass ein Bieter Quellprogramme oder Entwicklungsdokumente elektronisch lesbar oder in Papierkopie verschickt. Es ist aber ein gängiges Procedere – wie auch bei Wirtschaftsprüfern und Zertifizierungs-Auditoren, dass Einblick in jedes vom Prüfer gewünschte Teilprodukt gewährt wird. Die ausschreibende Institution wiederum muss bereit sein, einen gewissen Aufwand zu betreiben, *bevor* sie den Zuschlag erteilt und Verträge unterschreibt. Bei bewährten Standardprodukten reduziert sich dieser Aufwand erheblich, denn dort genügen Kunden-Referenzen.

4 Fazit

Ziel des Beitrags war es herauszuarbeiten, was man unter einem Campus-Management-System verstehen müsste. Es darf nicht der Versuch gemacht werden, alle betrieblichen Anwendungssysteme einer Hochschule dort hinein zu definieren, im Gegenteil: Das System lässt sich auf den Kern eines datenbankgestützten Systems fokussieren.

Der Kern eines Campus-Management-Systems unterstützt den Prozess *Lehre*, gestützt auf eine stets konsistente, transaktionssichere Datenbasis. Ein solches System existiert als Standard derzeit noch nicht, obwohl der Bedarf sehr groß ist. Auch Pilotinstallationen liegen in unterschiedlichem Umfang seitens der Anbieter, aber auch hinsichtlich deren Granularität vor. Dabei muss beachtet werden, dass auch Pilotsysteme ausgesprochen schwergewichtige Organisations- und Softwareprojekte sind.

Mit einer Checkliste am Schluss des Beitrags wird Hochschulen eine Hilfestellung gegeben, *was* bei einer Ausschreibung für ein Campus-Management-Systems zwingend beachtet und geprüft werden muss.

5 Literaturverzeichnis

[AR99] Appelrath, H.-J.; Ritter, J.: R/3-Einführung. Springer, Berlin et al. 1999.

[BB09] Bick, M.; Börgmann, K.: A Reference Model for the Evaluation of Information Systems for an Integrated Campus Management. EUNIS 2009, Santiago de Compostella, Spain, Sept. 2009. (die Quelle wird im Internet gefunden)

[Be09] Bensberg, F.: TCO-Analyse von Campus-Management-Systemen – Methodischer Bezugsrahmen und Softwareunterstützung. In: [HKF09], S. 493-502.

[Br+09] Brune, H.; Jablonski, M.; Möhle, V.; Spitta, T.; Teßmer, M: Ein Campus-Management-System als evolutionäre Entwicklung. In: [HKF09], S. 483-492.

[De79] Denert, E.: Software-Modularisierung. Informatik Spektrum, 2(1979) 4, S. 204-218.

[FH09] Fischer, H.; Hartau, C.: STiNE an der Universität Hamburg – Zur Einführung eines integrierten Campus Management Systems. In: [HKF09], S. 533-542.

[Fi99] Fischer, J.: Informationswirtschaft – Anwendungsmanagement. Oldenbourg, München - Wien 1999.

[Gr+10] Grechenig, T.; Bernhart, M.; Breiteneder, R.; Kappel, K.: Softwaretechnik – Mit Fallbeispielen aus realen Entwicklungsprojekten. Pearson Studium, München 2010.

[HKF09] Hansen, H.R.; Karagiannis, D.; Fill, H-G. (Hrsg.): Business Services – Konzepte, Technologien, Anwendungen (Bd 2). 9. Int. Tagung Wirtschaftsinformatik, Febr. 2009 Wien.

[HSU06] Helmut-Schmidt-Universität Hamburg: Empfehlung für die Auswahl eines Softwaresystems Prüfungsverwaltung / Campusmanagement. Sommer 2006. www.hsu-hh.de/campus-portal/index_V6lDrbV51OIcKbVm.html (wird auch ohne login gefunden: am 4.8.2010)

[HR09] Herzwurm, G.; Pietsch, W.: Management von IT-Produkten. dpunkt Heidelberg 2009.

[Kü97] Küpper, H.-U., Controlling, 2. Aufl. Schäffer-Poeschel, Stuttgart 1997.

[Me04] Mertens, P.: Integrierte Informationsverarbeitung, Bd 1. 14. Aufl. Gabler, Wiesbaden 2004.

[MV04] Mecklemburg-Vorpommern: Landeshochschul-Informationssysteme (2004 bis 2007) http://www.campusmv.de/index.php?id=publikationen (am 02.08.2010)

[Pa72] Parnas, D. C.: On the Criteria to be Used in Decomposing Systems into Modules. CACM 15(1972) 12, pp. 1053-1058.
[Ra09] Radenbach, W.: Integriertes Campus Management durch Verknüpfung spezialisierter Standardsoftware. In: [HKF09], S. 503-512.
[SB08] Spitta, T., Bick, M.: Informationswirtschaft. 2.Aufl., Springer, Berlin et al 2008.
[SG95] Schäfer, J. P.; Grauer, M. (Hrsg.): Universitätsverwaltung und Wirtschaftsinformatik. Proceedings, Siegen Okt. 1995.
[SK95] Sinz, E., Krumbiegel, J.: Gestaltung qualitätsgesicherter Universitätsprozesse am Beispiel des Prozesses 'Lehre und Studium'. In: [SG95], S. 15-32.
[SM95] Spitta, T.; Mordau, J.: Entwicklung und Ergebnisse eines allgemeingültigen Fachkonzeptes für die Prüfungsverwaltung an Hochschulen. In: [SG95], S. 128-147.
[Sp89] Spitta, T.: Software Engineering und Prototyping. Springer, Berlin et al. 1989.
[Sp97] Spitta, T.: Standardsoftware zur Verwaltung und Führung von Fakultäten. Eingeladener Vortrag GI-Jahrestagung `97, Univ. Bielefeld, Fakultät für Wirtschaftswissenschaften, Diskussionspapier Nr. 354, August 1997.
[Sp+10] Sprenger, J.; Klages, M.; Breitner, M. H.: Wirtschaftlichkeitsanalyse für die Auswahl, die Migration und den Betrieb eines Campus-Management-Systems. Wirtschaftsinformatik 52(2010) 4, S. 211-224.
[St+07] Stender, B.; Jablonski, M.; Brune. H.; Möhle, V.: Campus Management von der Hochschule aus gedacht, Wissenschaftsmanagement, (2007) 6, S. 19-26. s. auch: http://www.uni-bielefeld.de/bis/projekt/WiMa-Artikel (am 02.08.2010)
[TUM08] http://portal.mytum.de/pressestelle/meldungen/news_article.2008-01-21.7932373032 am: 08-07-2008. Aktuell: http://portal.mytum.de/iuk/cm/index_html (am 02.08.2010)
[TUM10] http://portal.mytum.de/iuk/cm/hintergrund/ (am 02.08.2010)
[UHH10] Uni Hamburg: http://www.info.stine.uni-hamburg.de/ (am 02.08.2010)
[We96] Weber, J.: Hochschulcontrolling – Das Modell WHU. Stuttgart 1996.
[Wiki10] Wikipedia: Amadeus Germany (Das START System). http://de.wikipedia.org/wiki/Amadeus_Germany (am 02.08.2010)